

# **Autix: Designing a Machine Learning Model to predict and maximize the value of modified and specialty vehicles**

**UTDiscovery Autix Project**

**Marcos Munoz, Mehedi Toufique, Sayema Rahman, Yaseen Mohammed**

The University of Texas at Dallas | CS 4475: Capstone Project

Mentor: Dr. Kemmeli | Dr. Chandra

## Table of Contents

<b>Table of Contents.....</b>	<b>2</b>
<b>Abstract.....</b>	<b>3</b>
<b>Objectives.....</b>	<b>3</b>
<b>Data Sources:.....</b>	<b>3</b>
<b>Data Fields.....</b>	<b>4</b>
<b>Car data from text using LLMs.....</b>	<b>5</b>
<b>Web Scraping Classic Cars.....</b>	<b>5</b>
<b>Data Cleaning.....</b>	<b>6</b>
<b>Data Analysis.....</b>	<b>7</b>
<b>Tuning Models.....</b>	<b>8</b>
<b>References.....</b>	<b>11</b>

## **Abstract**

In this project, we developed and validated a pricing model related to Autix's curated value metric algorithm known as xEstimate. This algorithm will take in a car's information such as maintenance history, performance modifications, cosmetic modifications, social sentiment, and some baseline information, and will give the user a price for their car based on the input received. Normally when a car owner wants to sell their vehicle, most online websites and apps don't consider some of the variables above. Hence, the owner gets a price that for them does not reflect everything that they have done to their car. AUTIX wants to provide its users with an efficient and precise algorithm that will provide an accurate estimated price for their vehicle considering more aspects. We hope to achieve the project goal by using Python and some of its libraries to help make/improve a working model. The code listed is in order.

## **Objectives**

1. Extract features from text data.
2. Web Scrape data from Classiccars.com
3. Develop a machine-learning model to predict car prices.
4. Use relevant features such as make, model, year, odometer reading, interior, and exterior color.
5. Achieve a high level of accuracy in predicting car prices.

## **Data Collection**

### **Data Sources:**

- Profiles
- ClassicCars(34000)
- ClassicCarsCleaned(900)

- Copy of classic\_data\_full
- Copy of dupon\_data\_full
- Copy of hemmings\_data\_full
- API dataset
- Started with a main data set provided by AUTIX which had all the important features that they wanted when using their ML model to predict a car's price. Every data set going forward would try to imitate the structure of the main data set.
- One of the main data sets was obtained by using web scraping on the car features description of a car in the Newest Listings section on ClassicCars.com.
- Later used three more data sets with car features from different websites and merged it into one data set.

### **Data Fields**

- Make
- Model
- Price
- Year
- Transmission
- Odometer
- Condition
- Engine type
- Age

### **Data Preprocessing**

- Handle the N/A, missing, or invalid values
- Format the values in each column - A lot of the values had different formats like
- Get rid of unnecessary symbols/values in each column
- Remove columns with little data
  - Trim
  - Interior and exterior colors
- Make a new column to categorize the cars by year
  - Antique: 1931-1975
  - Classic: 1976-1999
  - Normal:  $\geq 2000$
- Combine multiple data sets

## Car data from text using LLMs

### 🔗 GptAutixTextAnalysis.ipynb

- Code takes a data frame with a column with car descriptions df
- It accesses the get-API for it pre-built LLMs with an API key set to variable `open.api_key`
- We pass the car descriptions row by row with an inline for-loop to call the function `generate_columns`.
- The function uses a prompt you give it to give a reply with the `openai.Completion.create()`  
  
Reply is stripped for only a number then we add the generated column to the data frame.

### 🔗 LLamaLLMTextAnalysis.ipynb

- Car features available to get from the car description
- Using this API proved very effective in extracting desired data from text and putting it into a new column. It does have a small cost for each time it is prompted.

## Web Scraping Classic Cars

### 🔗 Webscrapautix-Version3.ipynb

- BeautifulSoup python library to get 35000 Car listings
- Made a python code to extract and access each car's URL link to get all the listed features and build a training data set that closely matches the profiles data set.

## Data Cleaning

### AutixClassicCarsDataCleaning

- Convert the "Price" column to float
- Remove "0" from the "Odometer" column
- Remove "Trim" and "Engine Size" columns
- Drop all rows containing N/A values
- Create a new column "Age" and add results after subtracting the "Year" column by 2023
- Create new column "label" which classifies the car by the "Year" column
  - 1931 to 1975 is antique
  - 1976 to 1999 is classic
  - 2000 and greater is modern

### AutixClassiccarsImputation.ipynb

- Delete columns that do not help with imputation target variable
  - "URL", "Description", "Trim", "VIN"
- Remove N/A value from "Price"
- Separate the data frame into features and target
- Train the RandomForestClassifier
- Predict the conditions for each row and include them into a cleaned data frame

### AutixMultipleCarsDataCleaning.ipynb

- Using Hemmings, Classic, Dupont Registry, and ClassicCars.com data
- Combine all the data into a data frame
- Convert the values in the "Price" column into float
- Extract miles from the "Odometer" column
- Create a new data frame with the "Price", "Make", "Model", "Year", "Transmission", "Odometer" and drop N/A values
- Create another data frame with "VIN", "Price", "Make", "Model", "Year", "Engine Size", "Transmission", "Odometer" and drop N/A values
- Create another data frame with "VIN", "Price", "Make", "Model", "Year", "Transmission", "Odometer" and drop N/A values
- Save all three data frames to a CSV file

### GettingModelDetails.ipynb

- Getting information from "Kaggle" so that we could fill up the missing rows.
- Merging both datasets and removing missing values.
- Make another attempt on the Kaggle dataset to fill up more rows, identify columns that have missing values, and merge all datasets to be consistent.
- Filter out the "model" and export to CSV.
- Removing unnecessary columns
- Fix data format
- Removing NA values


## Data Analysis

### Data analysis.ipynb

- Using our datasets from “hemmings”, “dupon”, “classiccars”, and “classic”; we merge to see how many missing values each column has.
- We do a general histogram of “Price” to see distribution.
- We changed the range of “Price” to see distribution and concluded that the data is more skewed while choosing big ranges but gets more evenly distributed by choosing a smaller range.



## Tuning Models

### AutixCorrelationModelAnalysis.ipynb

- Used the  Copy of ClassicCarsCleaned(900) data set to make and analyze ordinal categorical variables.
- Made new columns like Age and label to use in categorizing different cars based on their age, ex: Vintage, Antique, Classic, and Normal.
- Using the new and existing columns from the data set, we assigned numerical variables to label, Condition, Transmission, and Engine Type.
- Calculated the correlation between Price and the new respective columns made that have \_numeric in the name. Analyzed the calculated correlation values and discovered what variables have a good correlation with the price once assigned a numeric value.
- This helped in figuring out what car variable columns have an impact on the car Price.



### 🔗 PycaretModelSelectionClassiccars

- Imported necessary libraries for data analysis, preprocessing, visualization, and machine learning models.
- Loaded  Copy of ClassicCarsCleaned(900) from a Google spreadsheet, displaying the data types and basic data cleaning steps.
- Engineered new features like "Age" based on existing columns ("Year").
- Created a dictionary for mapping engine types to the number of cylinders and split the "Engine Size" column accordingly.
- Categorized vehicles based on manufactured year into "Vintage", "Antique", "Classic", and "Normal".
- Utilized PyCaret for regression analysis on a cleaned dataset, using models like XGBoost, tuning them for better performance, and evaluating them based on Mean Absolute Error.
- Best  $R^2 = 0.3$ , Best MAE = 30,000
- Processed another  Copy of ClassicCars(34000).xlsx performed similar data cleaning steps, and applied anomaly detection using the PyCaret library's AnomalyExperiment, specifically using a KNN model for anomaly detection.
- Removed data with high anomaly
- Filtered anomalies from the dataset and proceeded with regression analysis using the cleaned data, applying models like "Huber" regression, tuning them, and evaluating model performance.

- Best  $R^2 = -0.5$ , Best MAE = 37,000

#### 🔗 PycaretModelSelectionMultiplecars

- Imported necessary libraries and loaded a `Comb_filtered_with_modeldetails` and regression modeling.
- Performed initial data type checks and basic data cleaning operations.
- Created a new feature "Age" based on the "Year" column and visualized the distribution of prices using a histogram.
- Identified and removed outliers by employing Anomaly Detection using the KNN model.
- Saved the Anomaly Detection experiment for future reference.
- Split the dataset into two segments based on price ranges (below \$100,000 and below \$250,000) and perform regression analysis separately for these segments.
- Configured PyCaret's setup for regression modeling, including data preprocessing techniques like normalization, handling rare values, removing multicollinearity, and using PCA.
- Utilized PyCaret's `compare_models` function to select the best-performing models for both price segments (under \$100,000 and under \$250,000).
- Best for under \$100,000  $R^2 = 0.53$ , Best MAE = 10,000
- Best for under \$250,000  $R^2 = .48$ , Best MAE = 12,000
- Tuned the chosen models using `tune_model` and evaluated their performance using metrics like R-squared.
- Saved the best-tuned model using `save_model` for both price segments.

- Generated predictions using the trained models and visualized residuals using scatter plots to assess model performance.
- Made another regression setup to view feature importance without PCA.

## References

- <https://pypi.org/project/llama-cpp-python>
- <https://platform.openai.com/tokenizer>
- <https://textblob.readthedocs.io/en/dev/>
- <https://textblob.readthedocs.io/en/dev/>
- <https://pycaret.gitbook.io/docs/>